Use of USB Port as a Host for USB Key

Targeted competences: Use of USB in order to connect an USB key to the microcontroller

Hardware: STM32F7 Nucleo board

Framework: STM32 CubeMX 5.5.0 and Keil µvision V5.25.2.0

The aim of this document is to show how to use an USB port to connect a USB key to the microcontroller.

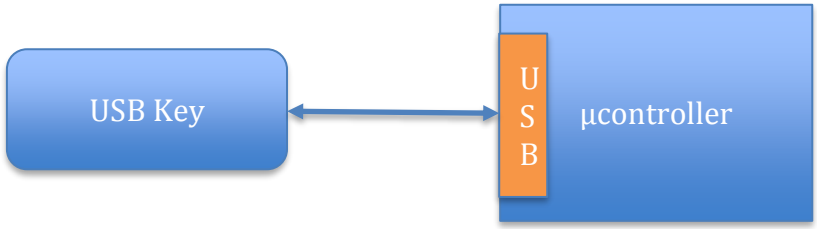

Figure 1: Global view of the system

We develop here a small code able to write a file on the usb key.

1. Microcontroller configuration

The first step is to configure the microcontroller. In our case, we use the NUCLEO-F767ZI platform based on a STM32F7 architecture. In order to configure this board we will use the CubeMx software. The first step after choosing the board is to name the project.

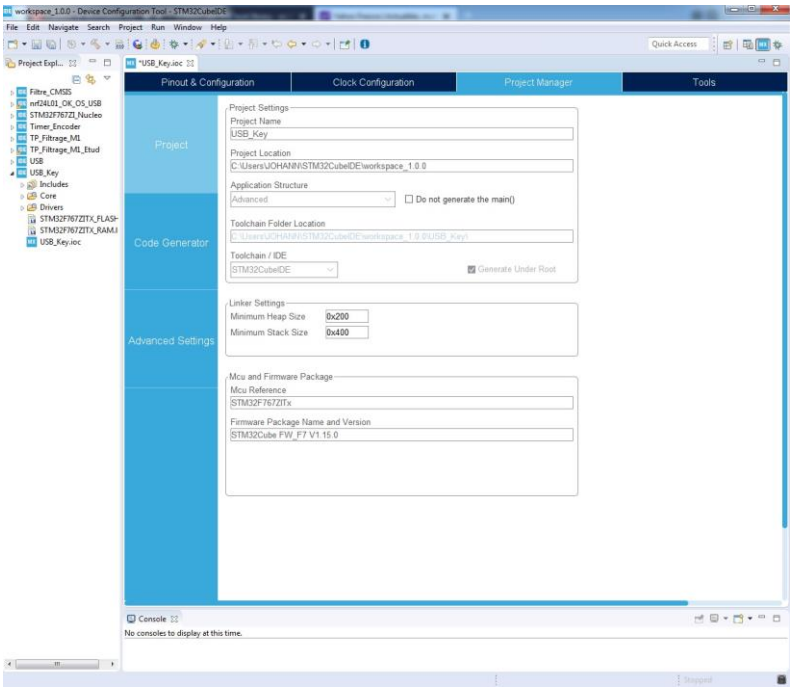

Figure 2:CubeIDE interface

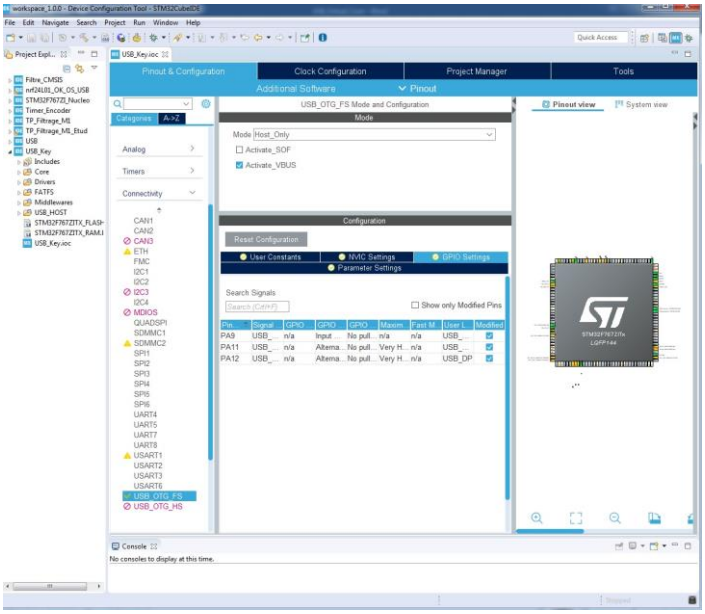

Figure 3: CubeMx interface to configure the STM32 Device

Here, I chose to use the USB_OTG_FS in host_only mode since we want to use an usb key as mass storage and no other peripheral that leads the need to use the usb port in device mode for instance. As we have to power the usb key, we need to set Activate_VBUS. Caution activate VBUS is not sufficient to power the usb key, we will have to write some code line to obtain the powering of the usb key.

The next step is to configure the usb in host mode. The figure below shows the configuration to realize.

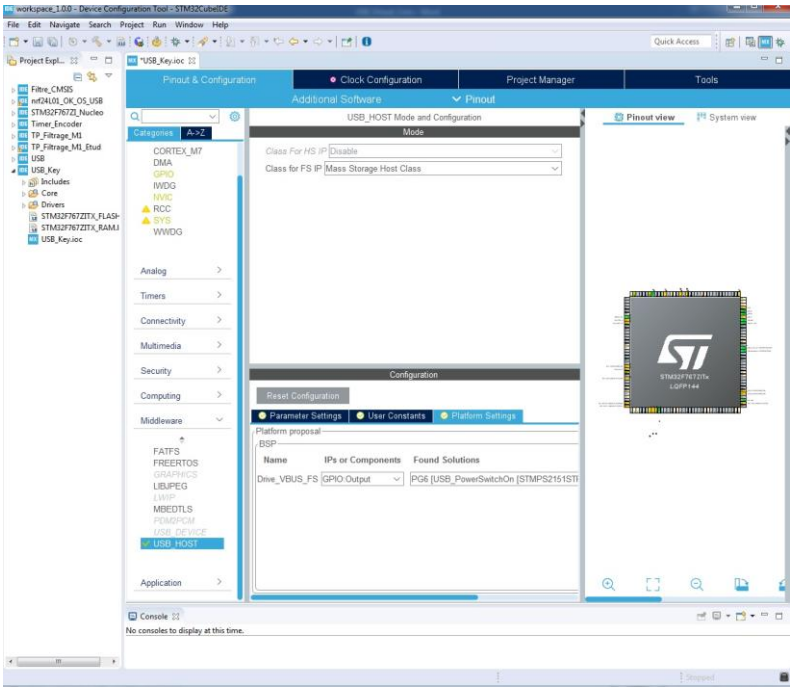

Figure 4: Host mode configuration

Here, we have to choose the class for the FS IP ; as we have chosen to use an usb key, we select the Mass Storage Host Class. The last step is the select the platform settings tab to choose a GPIO to drive the VBUS; we have to choose the PG6 pin.
Finally, the last step is to select the FATFS module in order to generate a file system for the USB Disk. Here I left the default settings.

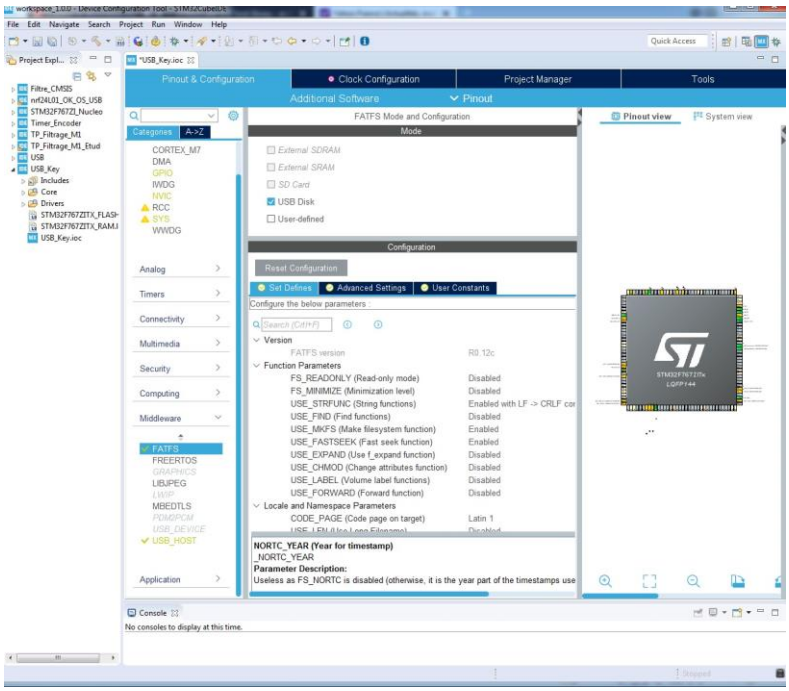

Figure 5: FATFS Configuration

Now we have to configure the clock for the whole board; to do this click on clock configuration item. With this configuration panel, we can choose the frequency of different µcontroller components as CPU frequency, AHB, APB1 buses and so on. I decide to use the maximum frequency of the SYSCLK that is 216MHz. In this case, the frequency of the APB1 and APB2 timer clocks are respectively 108MHz and 216MHz (see Figure 6).

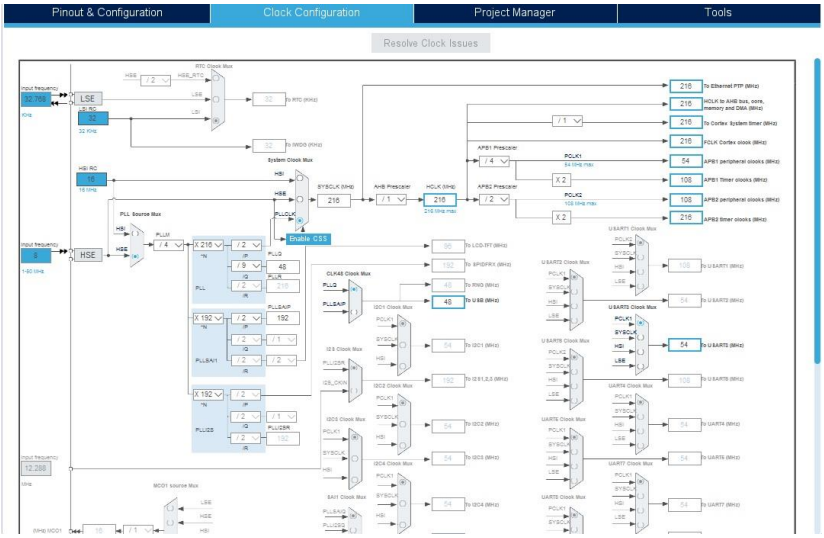

Figure 6: Clock configuration for the board

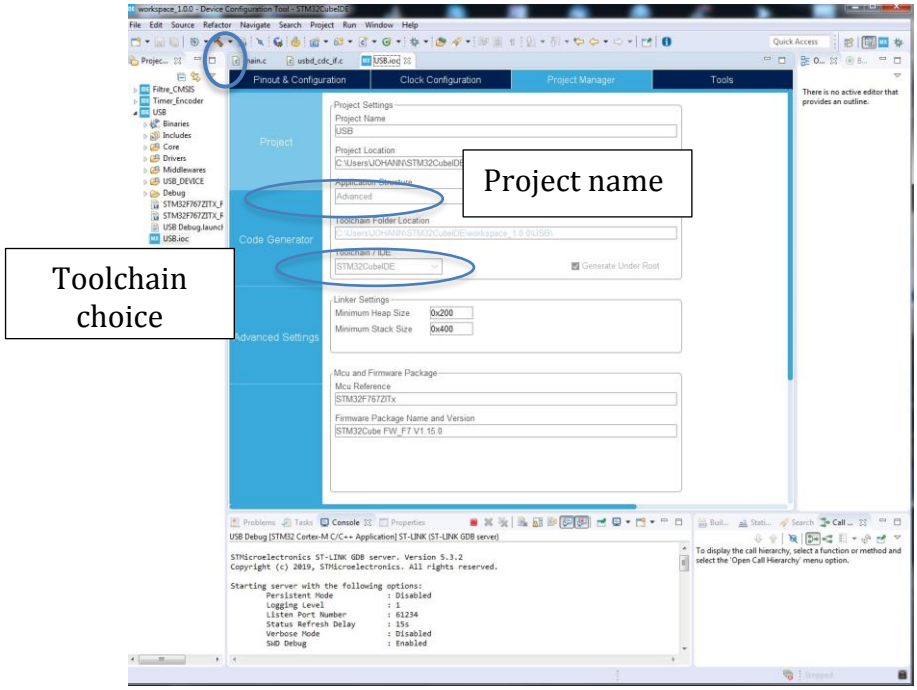

Figure 7: Project manager configuration

We have already given a name for our project and you can modify if you need the stack and heap parameters. Be careful if you use dynamic memory allocation indeed you should try to estimate the maximum memory size you need in order to choose the best heap size. If the heap size is too small, you will have some bugs during the program execution.

After clicking on Generate Code, CubeMx generates the application code and creates the project as shown in the figure below.

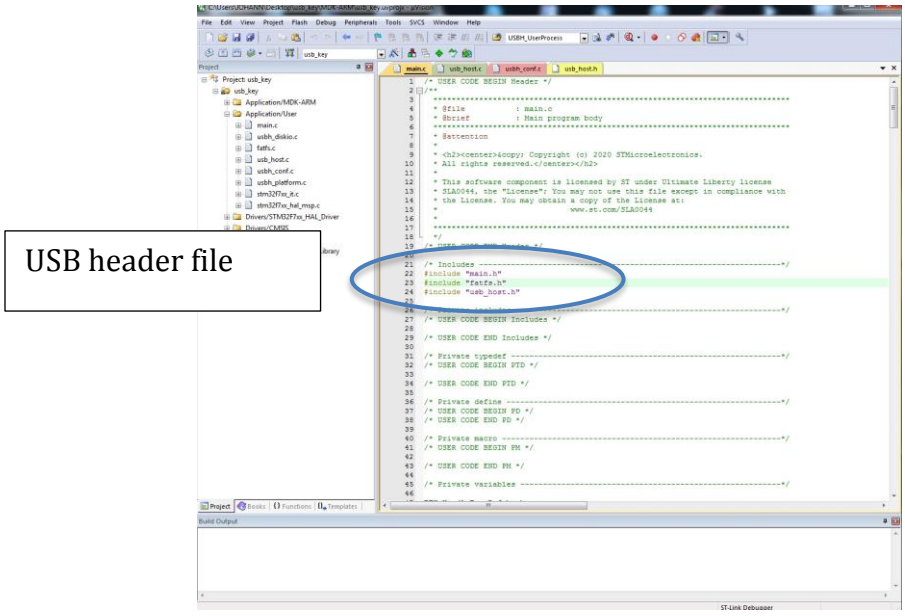

Figure 8: Code generation

In order to use the USB and FATFS APIs, we have to define some variables. I declared a file system object named USBDisk, a file object named FileToWrite, a variable

to store the path of the usb key and a variable to store the name of the file to write named NameOutputFile. Finally, I declared the Appli_state as an extern variable since this variable is defined in the usb_host.c file. I will use this variable in the main function to call the function Application when this variable is equal to Application_READY.

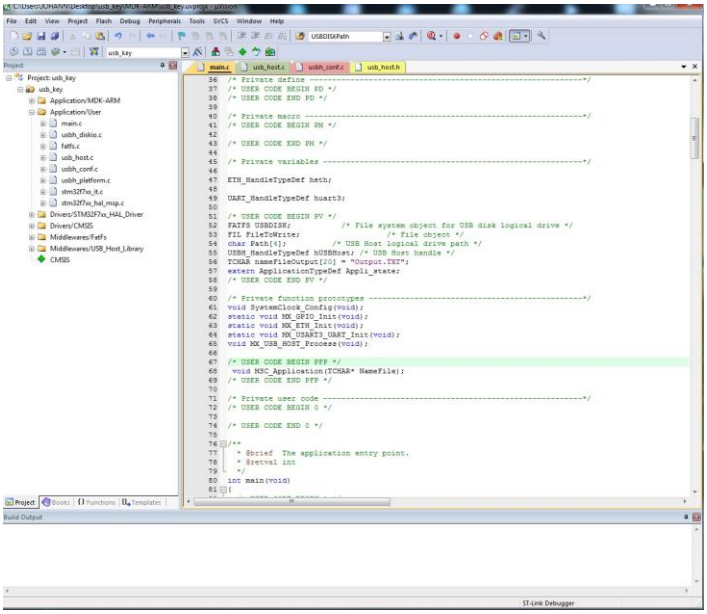

Figure 9: Variable declarations

In order to generate the VBUS signal to power the USB port, we have to modify the USBH_LL_DriverVBUS function since this function generated by the CubeMx tool is not functional.

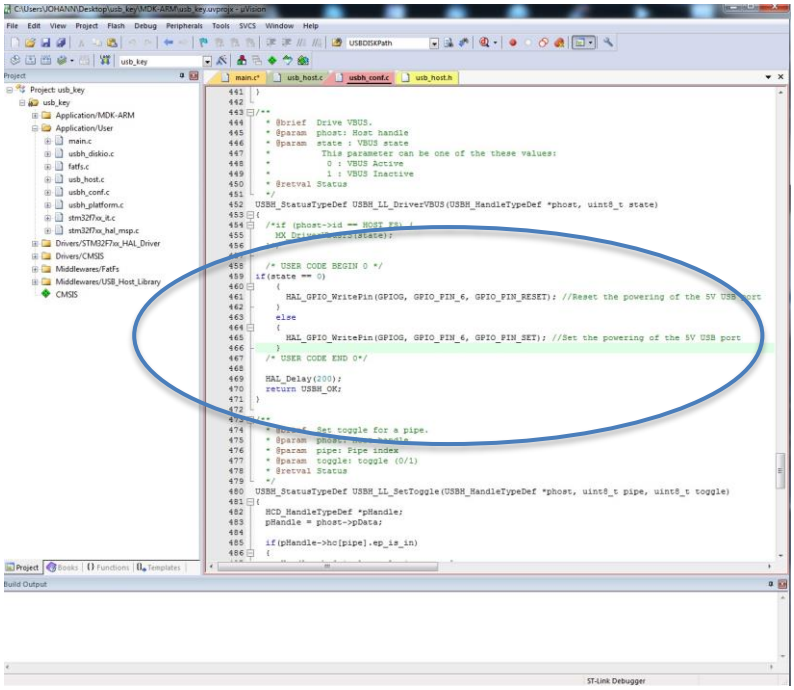

Figure 10: USBH_LL_Driver_DriverVBUS modification

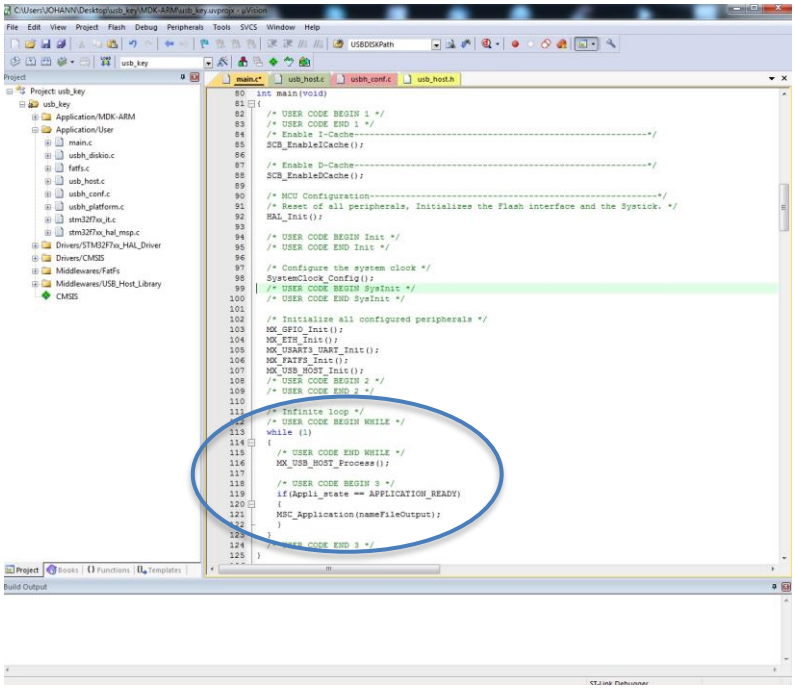

Figure 11: The main function code

In the While(1) part of the main.c I added some code in order to verify if the state machine of the usb core is in APPLICATION_READY state. In this state the usb port is ready to use so then I call the MSC_Application function with in parameter the name of the output file where I will write some text.

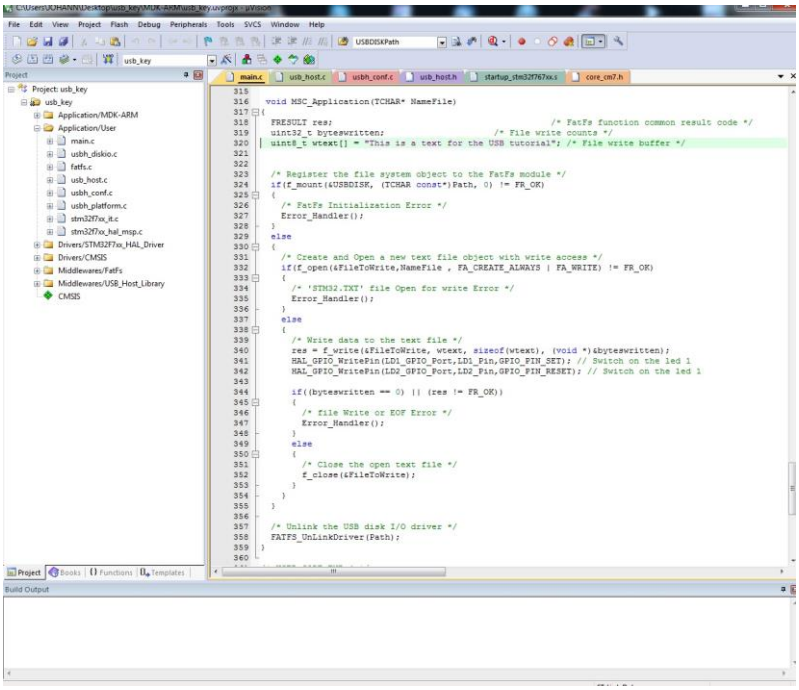

Figure 12: Code to write the text in the file in the USB key

At the end we have to compile the code in order to obtain the executable file for the STM32F7. When the compilation is done without error, we can transfer the code into the board.

A video showing the test of the code can be found here: https://youtu.be/43Rkx9ilfQo